Styling React Components

Antonio Ruberto
https://github.com/aruberto
https://www.linkedin.com/in/antonioruberto

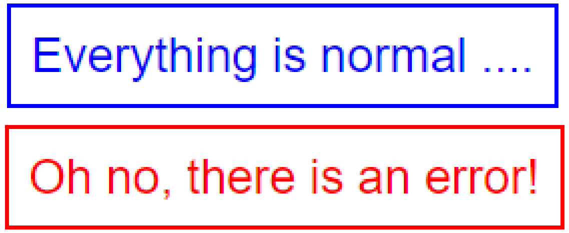
Introduction

- There are many approaches to styling React components.
- Review each approach to help you make an informed choice when developing your own components.
- Concepts can be directly applied developing components in other libraries/frameworks.

Introduction

- Demo each approach by building a simple Button:

```
<Button>Everything is normal ....</Button>
<Button error>Oh no, there is an error!</Button>
```



Everything is normal ....

Oh no, there is an error!

1. Old School Styling

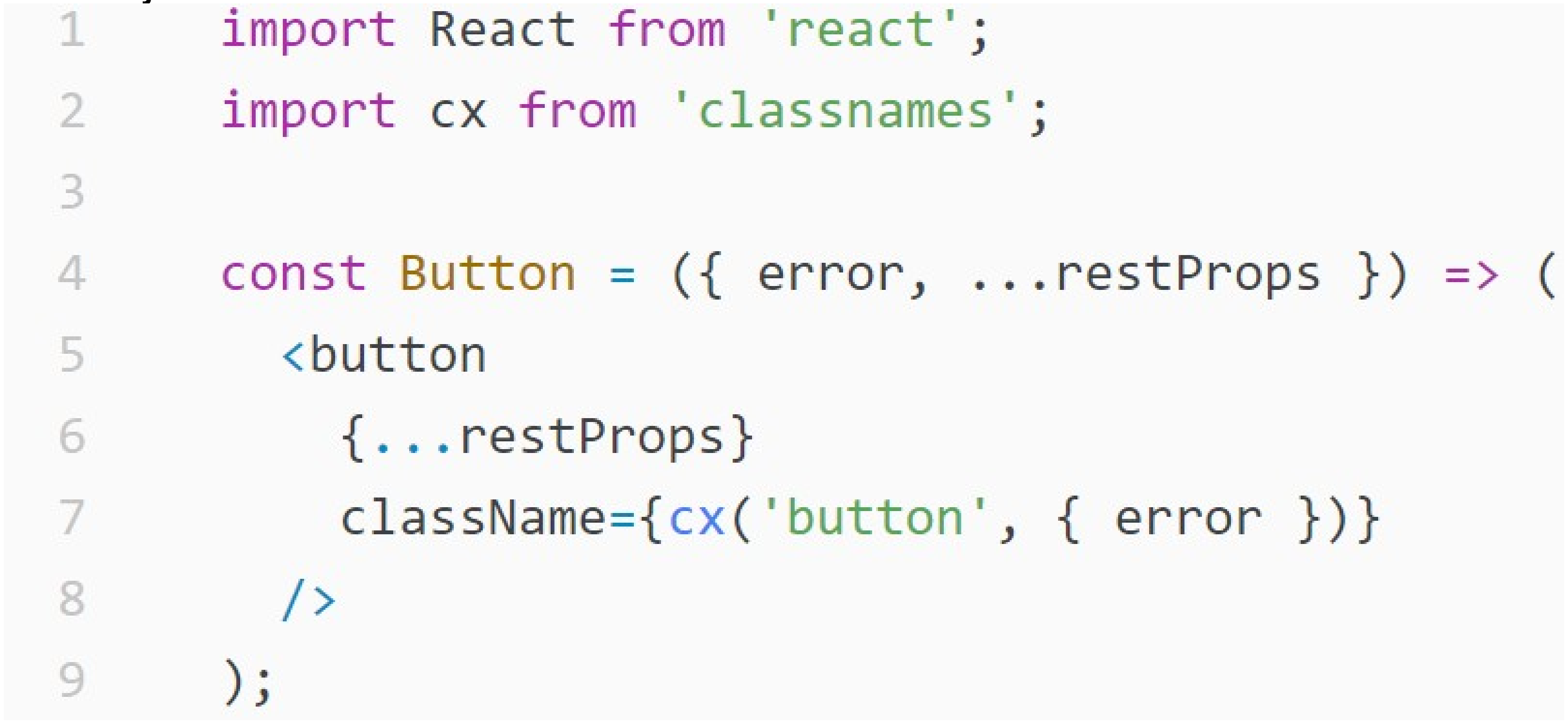
Old School Styling

1) Add some classes in your components

2) Set up CSS rules

Old School Styling

button.js

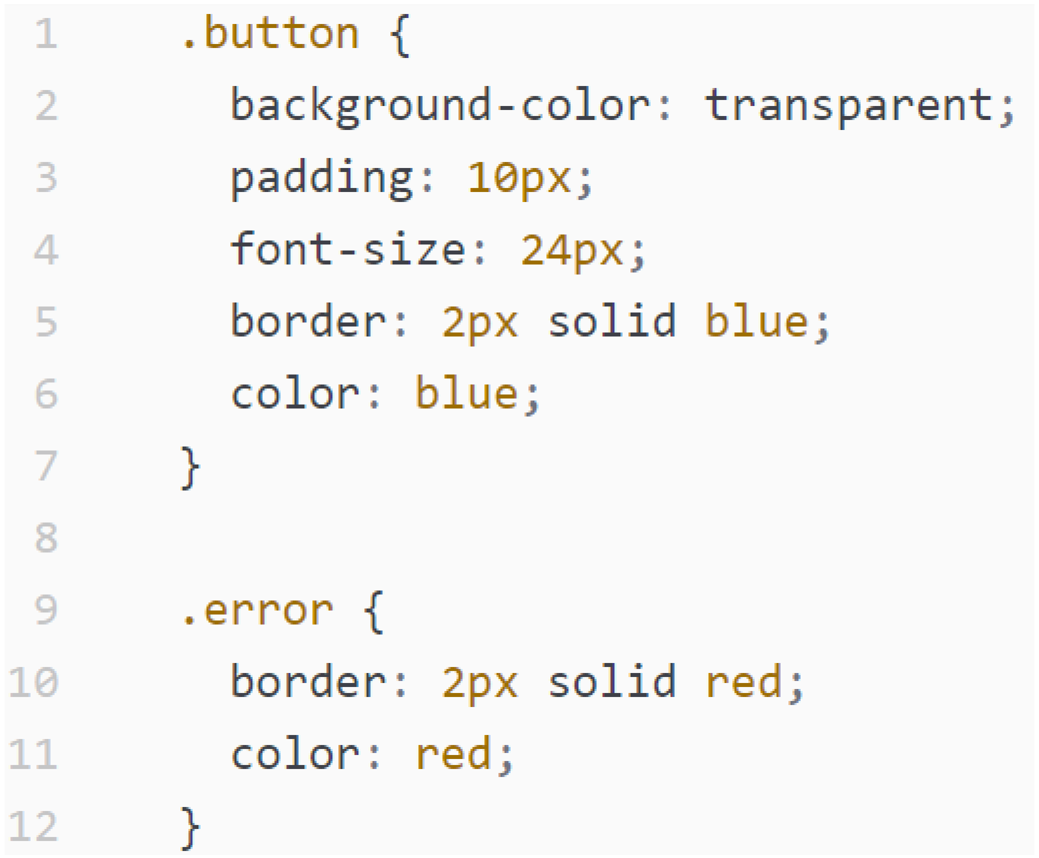


```
import React from 'react';
import cx from 'classnames';

const Button = ({ error, ...restProps }) => (
  <button
    {...restProps}
    className={cx('button', { error })}
  />
);
```

- { error } is ES2015 shorthand for { error: error }
- Using https://github.com/JedWatson/classnames to conditionally join CSS class names.
  - cx('button', { error }) returns 'button' when error is false, 'button error' when error is true.

Old School Styling

button.css



```css
.button {
  background-color: transparent;
  padding: 10px;
  font-size: 24px;
  border: 2px solid blue;
  color: blue;
}

.error {
  border: 2px solid red;
  color: red;
}
```

Old School Styling - Pros

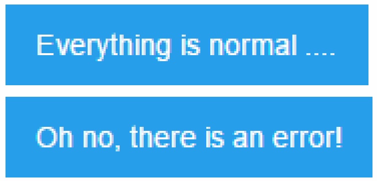
Old School Styling - Cons

- What if in future someone adds some more style sheets to your site ...

```
<link
  rel="stylesheet"
  type="text/css"
  href="https://cdnjs.cloudflare.com/ajax/libs/foundation/6.2.0/foundation.min.css">
```

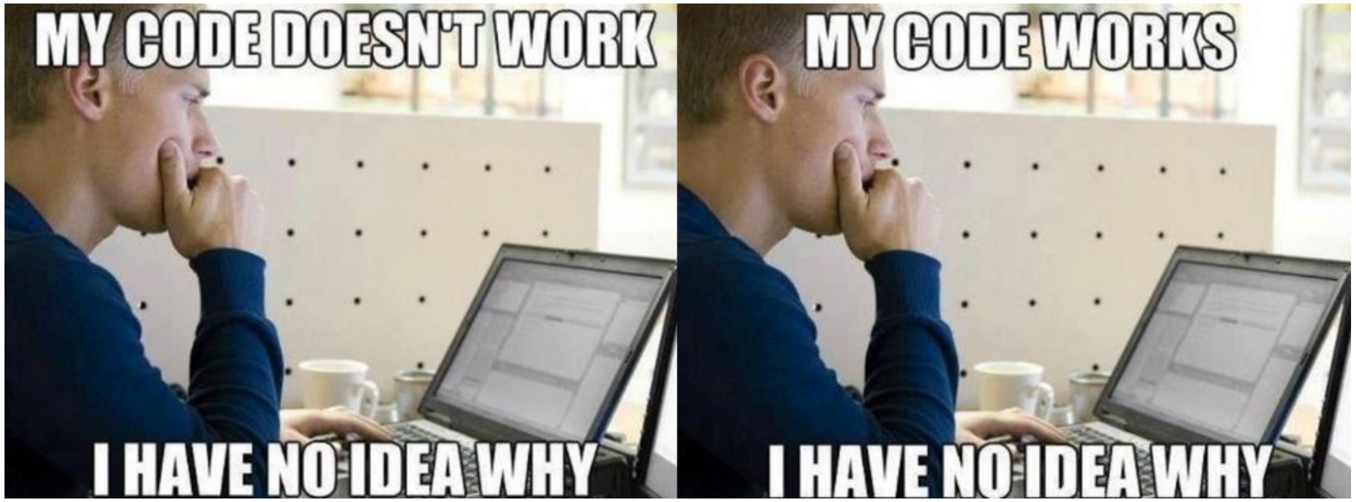
- Our Button component no longer looks right ...

```
<Button>Everything is normal ....</Button>
<Button error>Oh no, there is an error!</Button>
```



Old School Styling - Cons

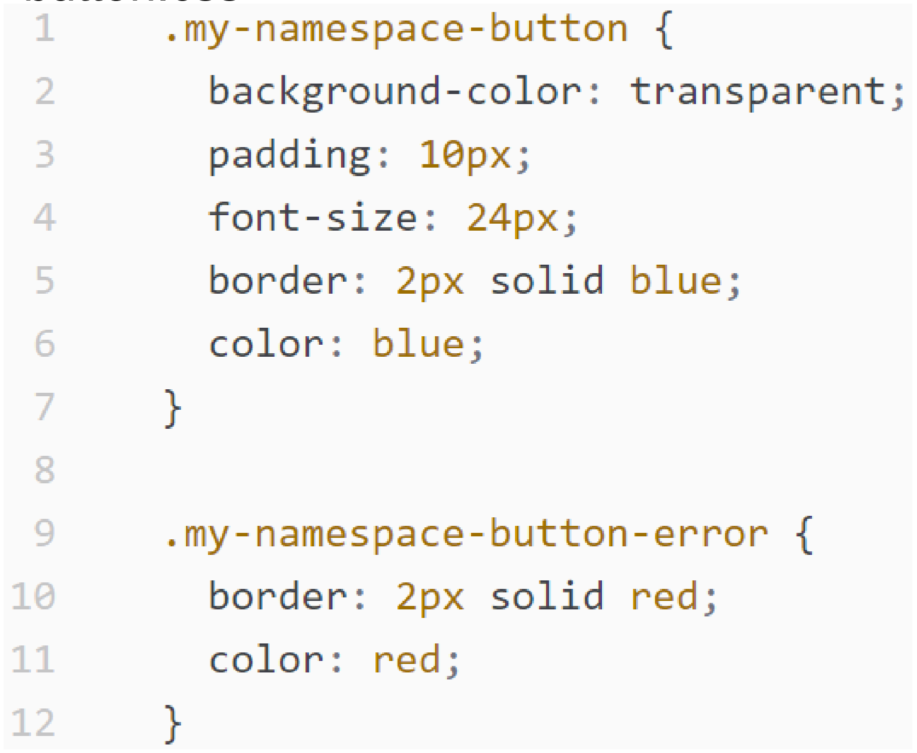
- In CSS everything is global.
- Button isn't a truly isolated component since the button and error CSS classes are essentially global mutable variables.
- As your application grows, it makes things hard to reason about.



Old School Styling - Cons

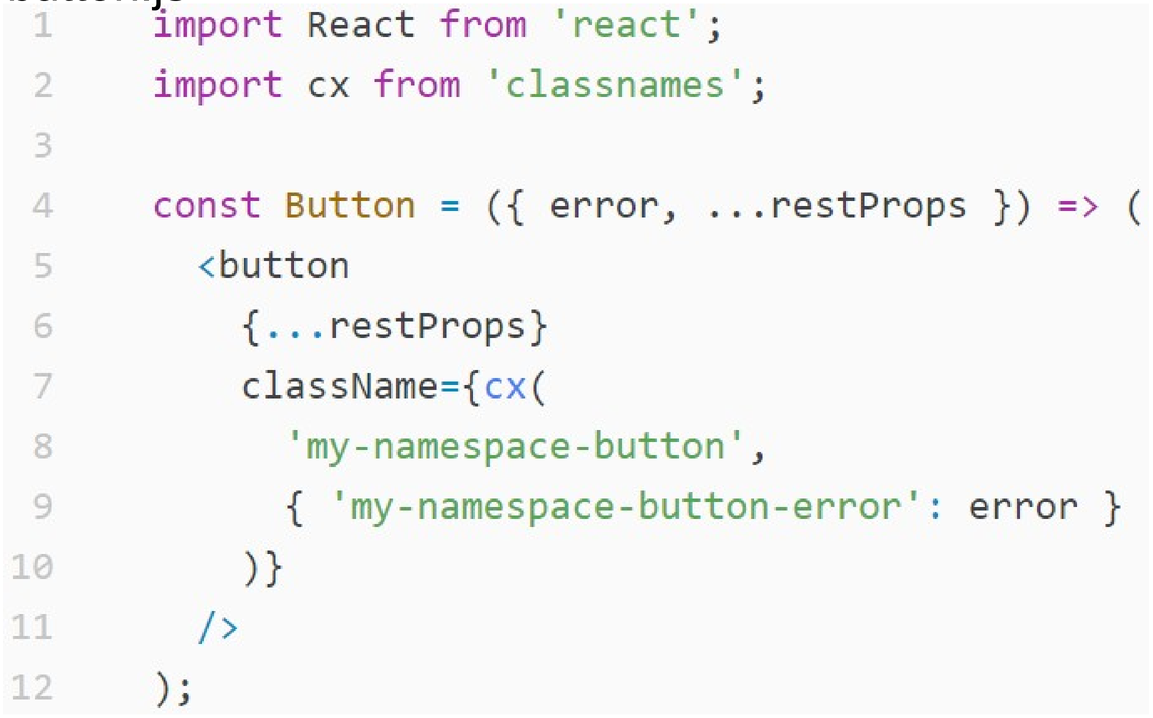
- Can partially solve this issue by name-spacing our class names (BEM, OOCSS, etc.) at cost of making our code more verbose.

button.css



```css
.my-namespace-button {
  background-color: transparent;
  padding: 10px;
  font-size: 24px;
  border: 2px solid blue;
  color: blue;
}

.my-namespace-button-error {
  border: 2px solid red;
  color: red;
}
```

button.js



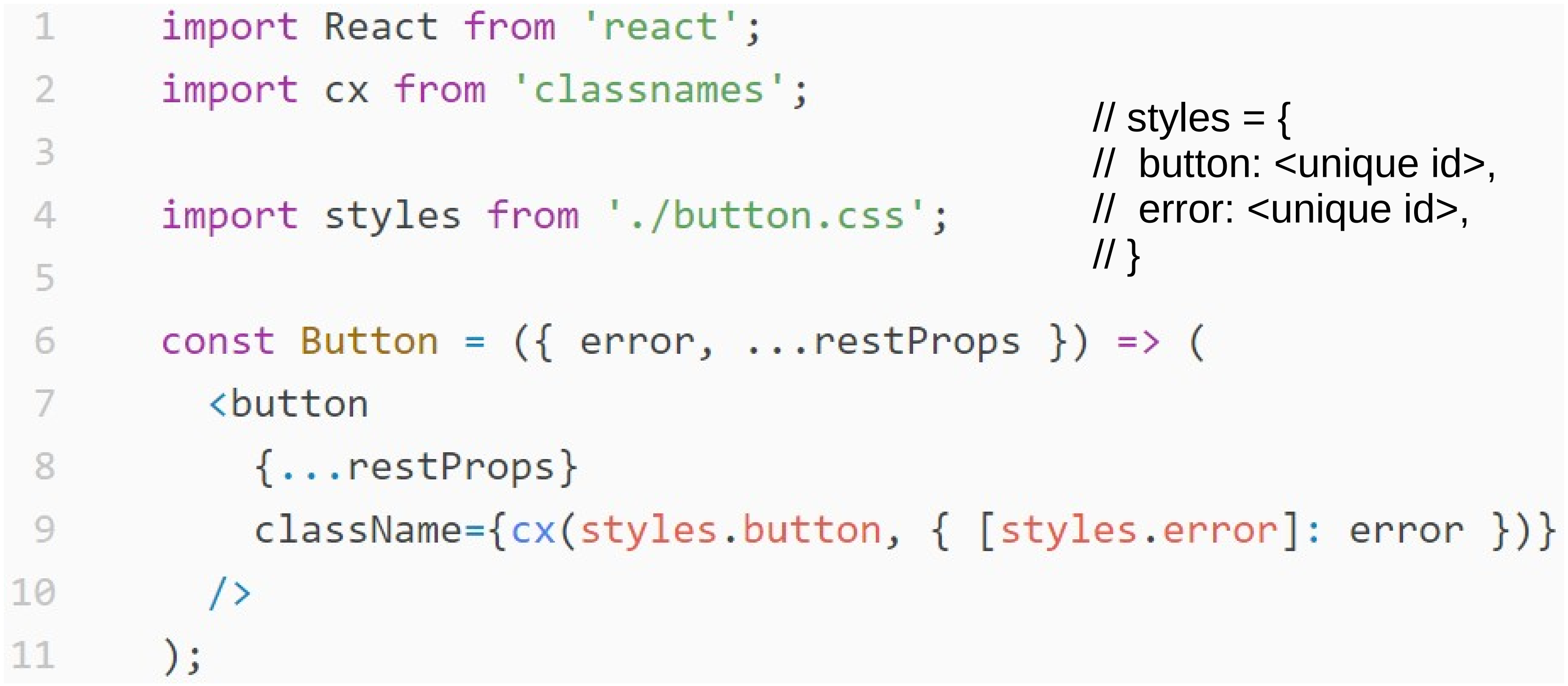
```js
import React from 'react';
import cx from 'classnames';

const Button = ({ error, ...restProps }) => (
  <button
    {...restProps}
    className={cx(
      'my-namespace-button',
      { 'my-namespace-button-error': error }
    )}
  />
);
```

2. CSS Modules

CSS Modules

- A CSS Module is a CSS file in which all class names are scoped locally by default.
- Component imports/requires CSS Module (just like you would any JavaScript dependency) and uses imported class names.

CSS Modules

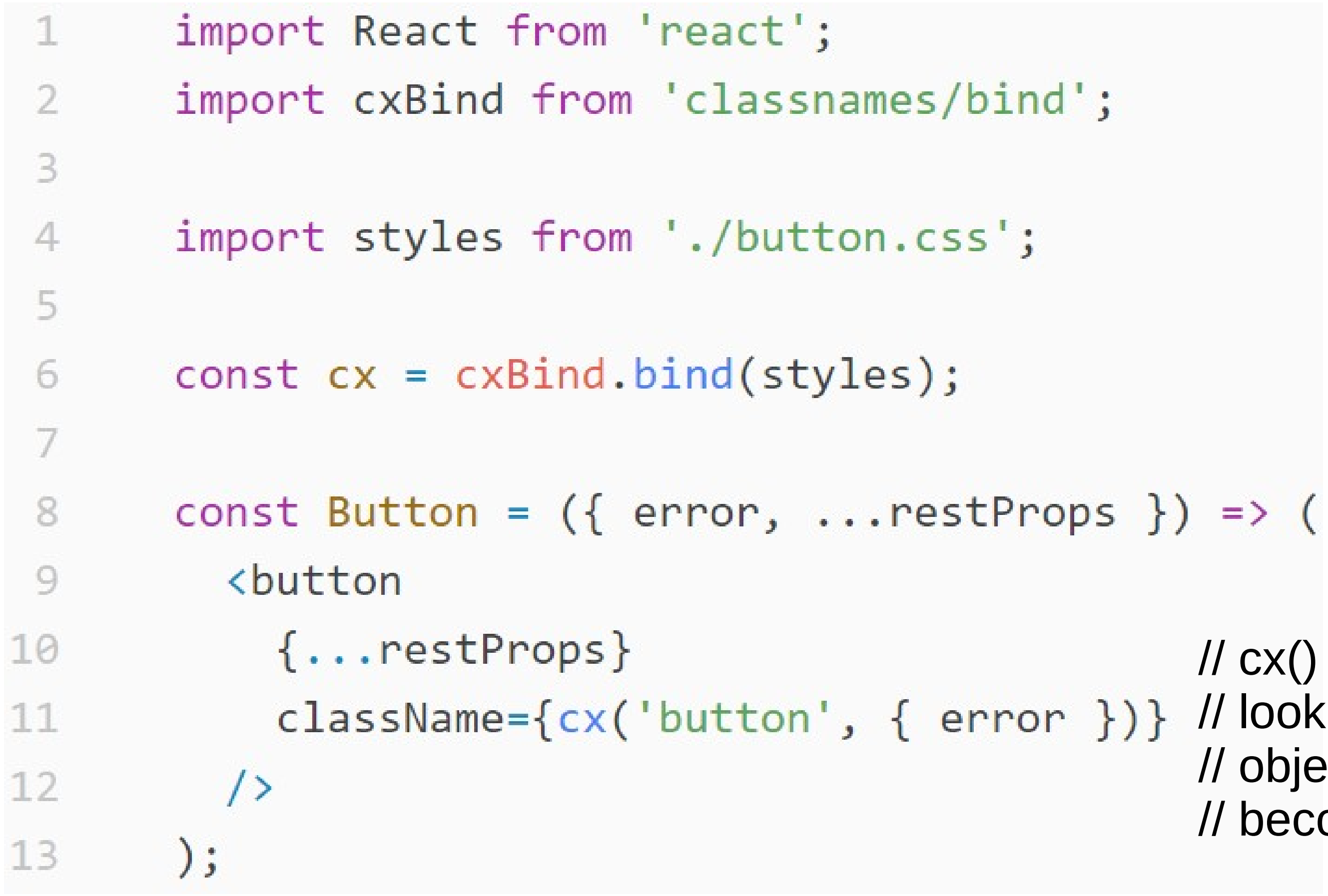


```
import React from 'react';
import cx from 'classnames';

import styles from './button.css';

const Button = ({ error, ...restProps }) => (
  <button
    {...restProps}
    className={cx(styles.button, { [styles.error]: error })}
  />
);
```

```
// styles = {
//  button: <unique id>,
//  error: <unique id>,
// }
```

CSS Modules

- Can make it even simpler using special “bind” version of classnames module designed for use with CSS Modules.



```
import React from 'react';
import cxBind from 'classnames/bind';

import styles from './button.css';

const cx = cxBind.bind(styles);

const Button = ({ error, ...restProps }) => (
  <button
    {...restProps}
    className={cx('button', { error })}
  />
);
```

```
// cx() performs
// lookup into styles
// object, ‘button’
// becomes styles.button
```

CSS Modules - Pros

- We can continue to use same CSS we know without having to worry about global conflicts.

CSS Modules - Cons

- Importing/requiring a CSS Module isn’t a standard require operation in a node CommonJS environment.
- Webpack (with css-loader) is only JavaScript bundler with non-experimental support for CSS Modules.
- Users are forced to use webpack, not a tool everyone has luxury of using.

webpack
MODULE BUNDLER

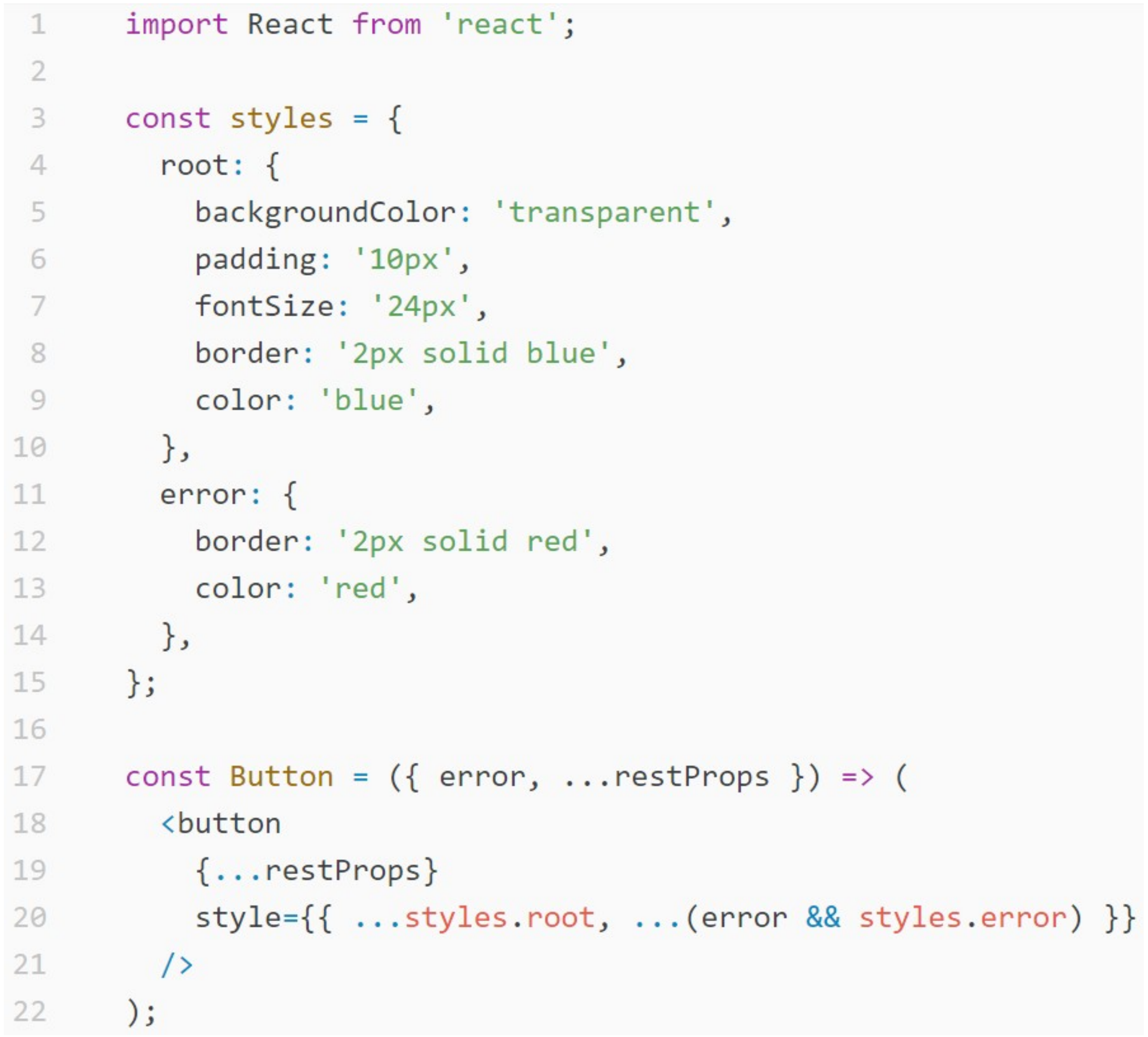
3. Inline Styles

Inline Styles

- Stop using CSS and simply set styles via style attribute/prop.
- In React, they are specified with an object whose key is the style name and whose value is the style's value.

Inline Styles

button.js



```
import React from 'react';

const styles = {
  root: {
    backgroundColor: 'transparent',
    padding: '10px',
    fontSize: '24px',
    border: '2px solid blue',
    color: 'blue',
  },
  error: {
    border: '2px solid red',
    color: 'red',
  },
};

const Button = ({ error, ...restProps }) => (
  <button
    {...restProps}
    style={{ ...styles.root, ...(error && styles.error) }}
  />
);
```

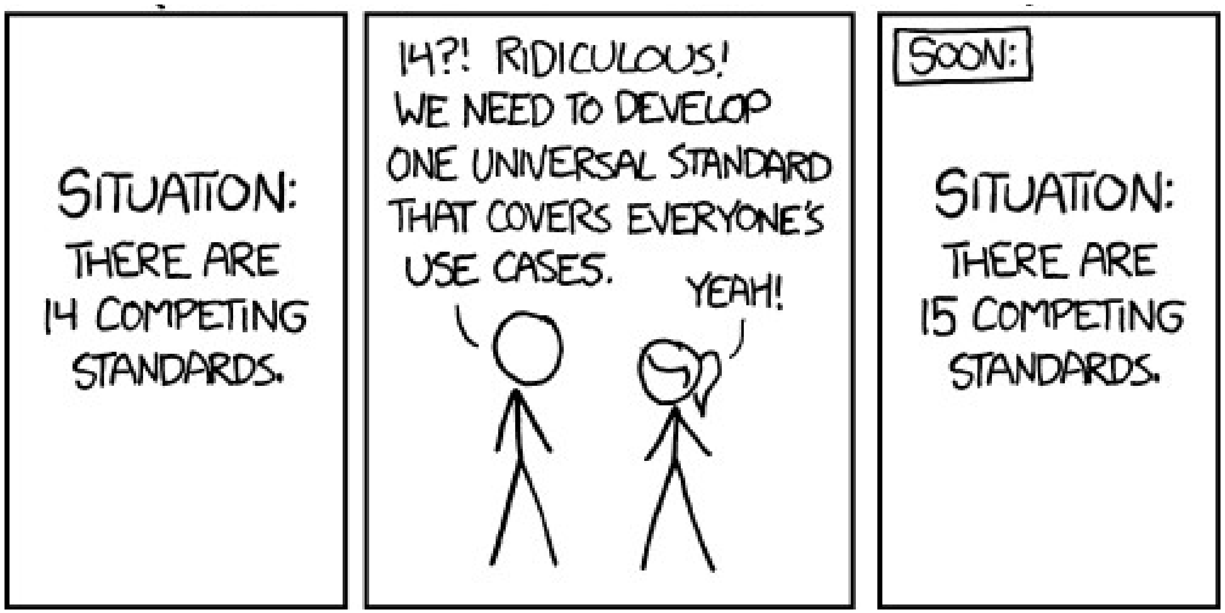
// Using spread (...)
// operator to merge
// styles.root and
// styles.error objects

Inline Styles - Pros

- Entire component source is now in a single file.
- Component user just needs to import component to use it:
  - No CSS to include or import
  - No CSS Module tooling to setup

Inline Styles - Pros

- One less tool/language. Use JavaScript instead of creating tools to make CSS more like JavaScript (SASS, LESS, PostCSS)


SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Inline Styles - Pros

- React Native doesn’t implement CSS.
- If planning on building both a web and native app, makes sense to use same approach for both.

Inline Styles - Cons

- Inline styles don’t cover all CSS features
  - Pseudo-selectors
  - Media queries
  - Keyframes for Animations

```
.button:hover {
  background-color: lightgrey;
}
```

VS

```
class Button extends Component {
  state = { hover: false };

  handleMouseOver = () => this.setState({ hover: true });

  handleMouseOut = () => this.setState({ hover: false });

  render() {
    const { error, ...restProps } = this.props;

    return (
      <button
        {...restProps}
        onMouseOver={this.handleMouseOver}
        onMouseOut={this.handleMouseOut}
        style={{
          ...styles.root,
          ...(error && styles.error),
          ...(this.state.hover && styles.hover),
        }}
      />
    );
  }
}
```

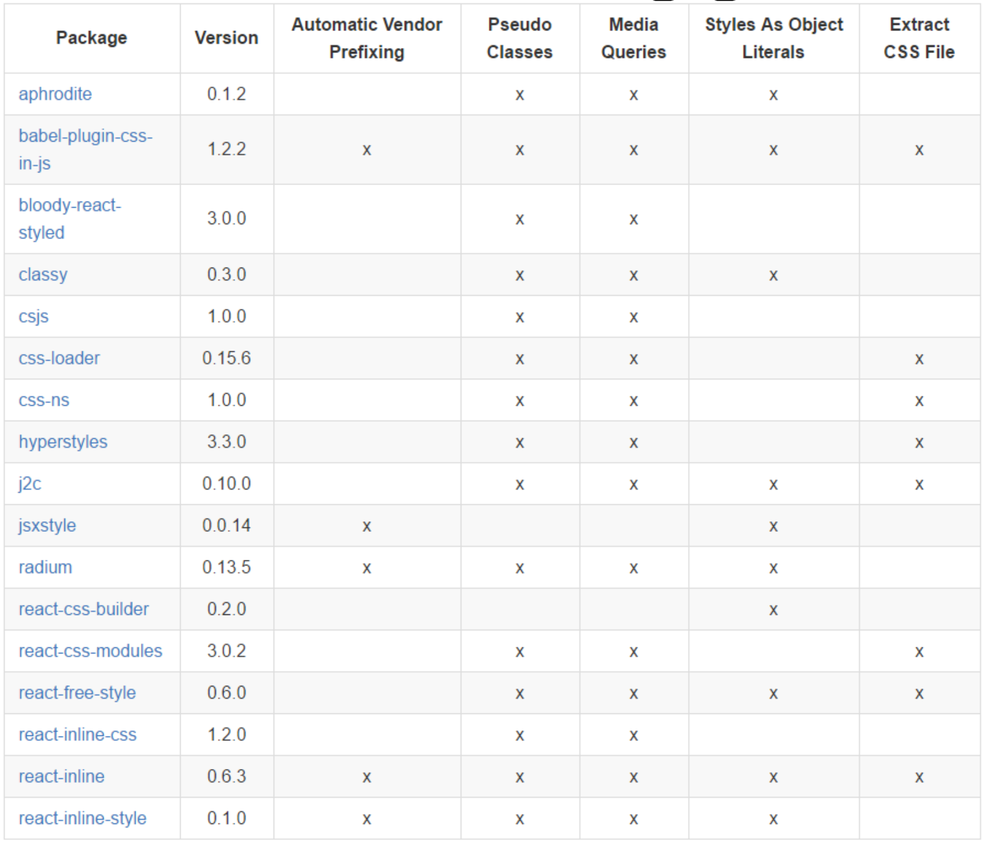
4. Future

Future

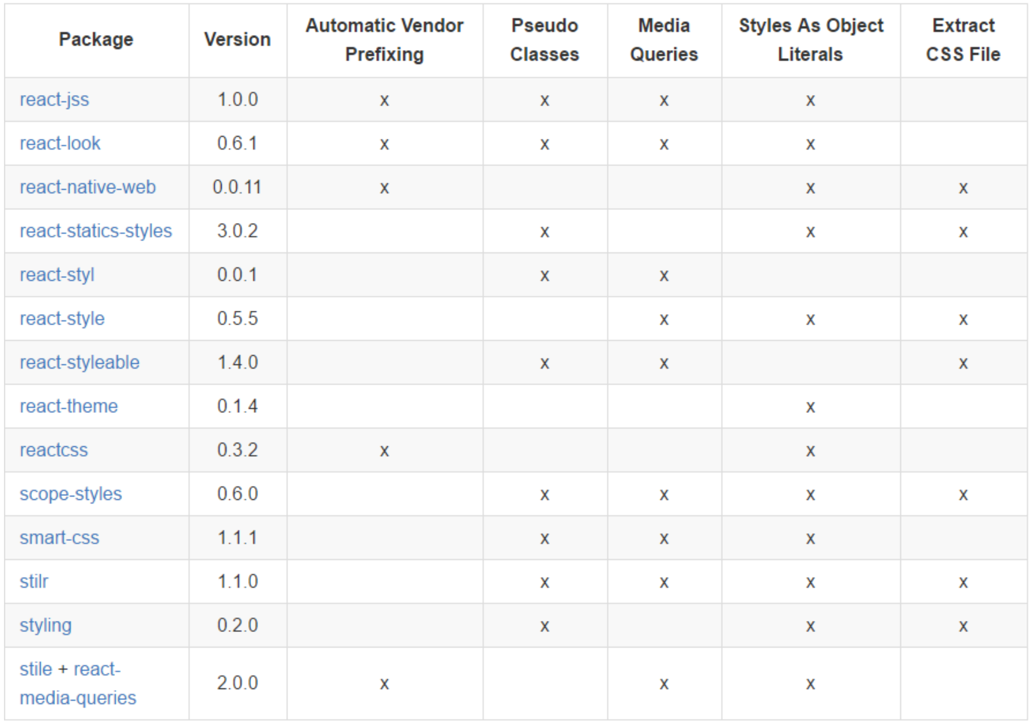
- Each styling approach has its pros and cons, there is no smoking gun.

Future

- 30+ libraries/tools released within last year trying to be that smoking gun



Package	Version	Automatic Vendor Prefixing	Pseudo Classes	Media Queries	Styles As Object Literals	Extract CSS File
aphrodite	0.1.2		x	x	x	
babel-plugin-css-in-js	1.2.2	x	x	x	x	x
bloody-react-styled	3.0.0		x	x		
classy	0.3.0		x	x	x	
csjs	1.0.0		x	x		
css-loader	0.15.6		x	x		x
css-ns	1.0.0		x	x		x
hyperstyles	3.3.0		x	x		x
j2c	0.10.0		x	x	x	x
jsxstyle	0.0.14	x			x	
radium	0.13.5	x	x	x	x	
react-css-builder	0.2.0				x	
react-css-modules	3.0.2		x	x		x
react-free-style	0.6.0		x	x	x	x
react-inline-css	1.2.0		x	x		
react-inline	0.6.3	x	x	x	x	x
react-inline-style	0.1.0	x	x	x	x	



Package	Version	Automatic Vendor Prefixing	Pseudo Classes	Media Queries	Styles As Object Literals	Extract CSS File
react-jss	1.0.0	x	x	x	x	
react-look	0.6.1	x	x	x	x	
react-native-web	0.0.11	x			x	x
react-statics-styles	3.0.2		x		x	x
react-styl	0.0.1		x	x		
react-style	0.5.5			x	x	x
react-styleable	1.4.0		x	x		x
react-theme	0.1.4				x	
reactcss	0.3.2	x			x	
scope-styles	0.6.0		x	x	x	x
smart-css	1.1.1		x	x	x	
stilr	1.1.0		x	x	x	x
styling	0.2.0		x		x	x
stile + react-media-queries	2.0.0	x		x	x	

- Example source and slides can be found at https://github.com/aruberto/styling-react-components
- Suggested Reading
  - CSS in JS by Christopher Chedeau
    - https://speakerdeck.com/vjeux/react-css-in-js
  - React: CSS in JS Techniques Comparison Michele Bertoli
    - https://github.com/MicheleBertoli/css-in-js
  - The Case for CSS Modules by Mark Dalgleish
    - http://markdalgleish.github.io/presentation-the-case-for-css-modules
  - Styling React by Juho Vepsäläinen
    - http://survivejs.com/webpack_react/styling_react